Java Method Browser

Gurvan Le Guernic
Lindstedtsvägen 3, level 4, office 1433
08 790 65 40
gurvan@kth.se

October 27, 2010

Abstract

The goal of the project is to develop a search engine for Java methods. The engine will not be based on keywords, but on distances between method signatures. The computation of distances between signatures relies on the inheritance hierarchy of Object Oriented programming languages. The project team will develop an Eclipse plugin that allows developers to query for a method signature. The plugin retrieves the "nearest" methods and displays the results in a way that reflect the distances between the query and the results.

1 Context

Programming languages offer an important number of libraries in order to facilitate developer's tasks. However, those libraries are huge. They contain an awfully big number of classes and an even bigger number of methods. Therefore, it is sometimes difficult to find the methods which fulfill the actions that we would like to carry out. Tools to help the developer find the right method exist. However, except for rare exceptions, these tools are only based on keyword search and present their results in a list-like fashion. For instance, Eclipse includes a search module based on keywords. Such solutions are only based on textual content of method descriptions and therefore lack a relation with the language semantics. For example, 'integer' is syntactically far from 'float', but semantically close. Additionally, list displays do not show the relevance of results or the similarities between them.

Previous research has proposed different ways to search for methods. One of them is based on a notion of distance between method signatures. Using this notion of distance, a method returning an 'integer' is said to be near methods returning a 'float' but far from methods returning a 'socket'. To search for a method, the user provide a method signature as close as possible to what he/she wants to do. For example, "String $\rightarrow$ InetAddress" is a potential request for a method that returns the IP address object associated to a machine name. The search engine will then return the methods whose signatures are the nearest from the query. This approach is similar to what some developers do manually when searching a method in the JavaDoc (the documentation of the the Java's Application Programming Interface (API) — library of functions designed to be used by developers in their applications).

The search engine to be implemented in this project will rely on a provided notion of distance for Java methods. It is expected to apply information visualization techniques to present the distance between the results and the query as accurately as possible. This representation will enable users to quickly perceive result relevance, as well as result similarities.

2 Prototype Requirements

The prototype should be available both as a standalone and as an Eclipse plugin adding, at least, a button to the Eclipse GUI to launch the search engine. The main window of the

search engine should contain:

- one or more text fields to enter the query under the form of a method signature,
- an area displaying the results in a list format including all necessary information,
- a 2D or 3D area displaying the results in a way reflecting the distances between the results and the query,
- an area allowing quick access to the Javadoc of a selected result.

As far as possible, the search engine should be able to handle a query in the `java` package by starting to display 4 or 5 results in 2 to 3 seconds. Ideally, the search engine would handle queries in the `java` and `javax` packages in 1 to 2 seconds.

3 Project Description

Project Team will be provided with a partial English translation of an article in French presenting the notion of distance used.

The project will involve developing:

- a piece of software to automatically extract inheritance data and method signatures of public classes from easily available sources,
- a piece of software, called data store from now on, to store the data collected and answer to queries effectively,
- a GUI compatible with the Eclipse's SWT library.

The main difficulty of the project will probably be the development of an efficient data store that is able to answer quickly enough to the signature queries. Nearly every method is an answer to a signature query, but only the nearest ones from the query signature must be returned. The distance from a method to a query is different for every query signature. Therefore this distance can not be completely precomputed when data is collected for method and classes. The distance will have to be at least partially computed during the query processing. The data store could be implemented using an ad hoc data structure or a generic embedded database, such as Derby, Oracle/Berkley DB or SQLite.

As the search engine handles Java methods, it would seems natural to use the Java language to implement it. However, other languages can be used. It is expected that the resulting software would be available under an open source license, but this point also can be discussed.

I am currently a postdoc researcher in the TCS group at KTH and will be able to provide assistance to the team involved in the project. I have some experience with such implementations and will be able to propose some potential solutions if the project team gets stuck somewhere. I will be able to devote approximately 1 to 2 hours per week to the team involved with the project.